Capitulo 1: Variáveis, tipos e operadores

Trabalhando com variáveis

Basicamente varáveis têm 3 momentos:

a. declaração (“batiza a variável e ela, então, passa a existir)

b. inicialização (atribui valor inicial)

c. uso (lê-se ou atualiza-se seu valor)

- Exemplo:

```
let idade;  //declara a variável “idade”
idade = 12;   //inicializa
console.log(idade)   //usa
let nome = “Zé”;  //declara e inicializa ao mesmo tempo
```

“Batizando” uma variável

Nomes de variáveis são chamados de *identificadores*

Identificadores são, na realidade, uma sequência de caracteres no código, que identifica uma *variável, função, ou propriedade de objetos*.

As regras para identificadores legais são praticamente as mesmas na maioria das linguagens de programação:

- podem conter somente caracteres alfanuméricos, "$" ou "_"
- Não podem iniciar com um dígito (0-9)

Um identificador difere de uma string no sentido de que uma string é informação, enquanto um identificador é parte do código.

Diferenças entre let e var

No exemplo anterior vimos que variáveis eram criadas por meio da palavra reservada "let". Em Javascript, além de *let*, pode-se usar *var e const* para se declarar uma variável:

```
var x = 10;

const y = 20;
```

Antes de 2015, *var* era a única maneira de se declarar uma variável;

***let* e *const* apareceram na versão ES2015**

Mas há várias diferenças entre *var, const* e *let*, como veremos nos slides a seguir.

Diferenças entre let e var: Redeclaração

Redeclaração:

var é possível declarar duas ou mais vezes uma mesma variável (ou seja, com o mesmo identificador)

```
var oi = "Oi!";
var oi = "Olá!";   // Sem problema
```

Com *let* não é possível

```
let tchau = "Tchau!";
let tchau = "Até!";   //SyntaxError
```

Com let não se pode redeclarar, mas pode-se reatribuir valor

```
let nome = “Maria”
nome = “Mara”
```

Diferenças entre let e var: escopo

Escopo:

var: escopo de função;

let: escopo de bloco;

Esse foi um dos principais motivos para a introdução do *let* em 2015; escopo de função era uma das grandes fontes de bug no Javascript

```
function test()
{
  for (var key in meuObjeto)
  {
    var foo = 'bar';
  }
}
//é equivalente a:
function test()
{
  var foo, key;
  for (key in meuObjeto)
  {
    foo = 'bar';
  }
}
```

Diferenças entre let e var: escopo

Escopo:

var: escopo de função;

let: escopo de bloco;

Esse foi um dos principais motivos para a introdução do *let* em 2015; escopo de função era uma das grandes fontes de bug no Javascript

```
for (let i = 0; i < 5; i++) {
  // i acessível ✔
}
// i não acessivel ✕
```

```
for (var i = 0; i < 5; i++) {
  // i acessível ✔
}
// i acessível ✔
```

Constantes

Declaradas por meio da palavra reservada *const*

Só apareceu no Javascript em 2015 (ES2015)

Variáveis definidas via const se comportam como let, exceto que não podem ser reatribuídas

```
const PI = 3.141592653589793;

PI = 3.14;      //gera erro

PI = PI + 10;   //também gera erro
```

Resumo: var x let x const

	block scope	binds to this	hoisted	allow redeclaration	allow reintialization
var	no	if global	yes	yes	yes
let	yes	no	no	no	yes
const	yes	no	no	no	no

Tipos de variáveis

- Tipos de dados definem os possíveis *valores* que podem ser representados e suas respectivas operações

- **tipos primitivos:**
  - string
  - number
  - boolean
  - Além desses, JS possui ainda os tipos primitivos: undefined, null, symbol e bigint

- **3 tipos complexos*:**

  object

  array

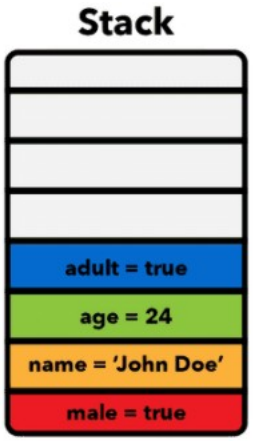
  function

  *na realidade, tudo é object!

Tipos primitivos vs tipos referência

Tipos primitivos:

```
const male = true
const name = 'John Doe'
const age = 24
const adult = true
```



Tipos referência:

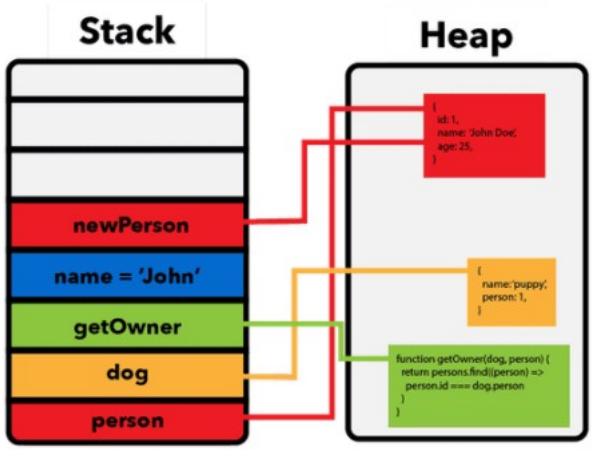
```
const person = {
  id: 1,
  name: 'John',
  age: 25,
}

const dog = {
  name: 'puppy',
  personId: 1,
}

function getOwner(dog, persons) {
  return persons.find((person) =>
    person.id === dog.person
  )
}

const name = 'John';

const newPerson = person;
```



Tipagem dinâmica

linguagens digitadas dinamicamente realizam verificação de tipo em tempo de execução (ao contrário de linguagens tipadas estaticamente, que realizam verificação de tipo em tempo de compilação):

- linguagens tipadas dinamicamente *não* exigem que você declare os tipos de dados de suas variáveis antes de usá-las

- uma mesma variável pode ser usada para armazenar valores de diferentes tipos de dado

```
let x = 10;                          //não precisa declarar tipo
x = “agora sou uma string”;          //mesma variável, diferentes tipos
```

Number

Números inteiros ou decimais

```
let idade = 42;          //não se declara o tipo das variáveis, ao
let pi = 3.14;           // contrário de outras linguagens (como Java, C, etc.)
```

Apenas um tipo numérico? 😱

Java: int, byte, short, long, double, float!

Number

Operações básicas: +, -, *, /, %

Precedência: igual às das aulas de matemática

Parênteses têm precedência maior do que os operadores e pode ser usado para mudar a ordem de execução:

3 + 2 / 4 -1 é diferente de (3+2)/(4-1)

String

Usada para armazenar textos

Valores são textos envolvidos por aspas simples ou duplas

let nome = “Zé da Silva”

let nome = Zé da Silva; //errado!

Podem ser concatenadas (“somadas”)

let nome = “Zé” + “da Silva”;

let frase = “O “+nome+” é meu amigo”; //utilidade real da concatenação!

Caracter de escape: “\”

let frase = ‘You\’ve got a friend in me’;

Boolean

Tipos de dados que aceitam dois valores: *true* e *false*

Geralmente usados em condições de teste (envolvendo expressões lógicas)

```
let casado = true;
let maior = idade > 17;
```

Expressão lógica
(ou booleana)

Array

Armazena múltiplos valores

Seu literal consiste nos valores separados por vírgulas e envoltos por colchetes

Valores são acessados a partir de seus *índices* (números inteiros que indicam a posição de cada valor dentro do array)

```
let meuArray = [“Maria”, “Pedro”, “Lucas”]
MeuArray[0]
```

Iterando em um array

Uma das operações mais comuns que se faz com um array é executar uma dada operação repetidamente em todos seus elementos (o que chamamos de iterar):

```
let sequencia = [1, 1, 2, 3, 5, 8, 13];
1) for (let  i = 0; i < sequencia.length; i++) {          //for tradicional
        console.log(sequencia[i]);
   }
2) for (const i of sequencia) console.log(i);          //for... of
3) sequencia.forEach(x=>console.log(x))                //forEach
```

Exercícios

1) crie um array com seus cantores preferidos

2)imprima seu conteúdo na tela usando o for “tradicional”

3)imprima seu conteúdo na tela usando o “for... of”

4)imprima seu conteúdo na tela usando o forEach()

Objeto

Modelam objetos da vida real

Tecnicamente, objetos são arrays associativos (também conhecidos por map, dictionary, hash table, symbol table, etc)

- Coleções de pares *(chave, valor)*, onde chave é uma string e deve ser única

Valores do objeto são acessados usando a *notação ponto*

```
let aluno = { nome: ”Carlos”,
              Matricula: 1234 }  //objeto literal
console.log(aluno.nome)      //acessando valor do objeto: notação ponto
console.log(aluno[“nome”])   //notação alternativa
```

Objeto

O tipo Objeto (Object) é especial.

Todos os outros tipos são chamados de “primitivos” porque seus valores podem conter apenas uma única coisa (seja uma string, um número ou qualquer outro). Em contraste, os objetos são usados para armazenar coleções de dados e entidades mais complexas.

Por serem tão importantes, os objetos merecem um tratamento especial. Trataremos deles mais tarde no capítulo Objetos

Literais Javascript

Inteiros:

Decimais: 10, -35

Octais: 017, -035

Hexadecimais: 0xAF, -0x89

Float:

78.90, 7.89e1

String:

'alo, mundo',

"alo, mundo"

Array:

[1, 2, 3]

Boolean:

true, false

Object:

{nome: "Zé", sobrenome:"da Silva"}

Regex:

/ab+c/

Dois tipos especiais

Null

Variável não aponta para objeto nenhum

Undefined

Variável ainda não foi atribuído valor

Non-zero value

null

0

undefined

Exercícios

Use, para as tarefas a seguir, editores online como codepen.io, jsfiddle.net, glitch.com ou repl.it

1) pede-se que você:

Declare uma variável chamada meuNome.

Inicialize meuNome com um valor adequado, em uma linha separada (você pode usar seu nome real ou qualquer outra coisa).

Declare uma variável chamada minhaIdade e inicialize-a com um valor, na mesma linha.

2) Usando console.log()

Use a função acima para imprimir as duas variáveis da questão anterior.

Exercícios

3) O que acontece se somarmos um número a uma string?

"Maria"+ 243 //dá algum erro? Se não, o que acontece?

4) É possível converter um número em uma string? Como? Funciona com literais? Como mudar a base (p.ex. Hexadecimal)?

5) É possível converter uma string em um número? Como?

6) Qual a diferença entre tipagem estática e dinâmica? Javascript implementa que tipagem?